VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Nástroj pro vizuální modelovaní cílů

Goal oriented Visual Modeling Tool

2018 Adam Maják

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student:	**Adam Maják**
Studijní program:	B2647 Informační a komunikační technologie
Studijní obor:	2612R025 Informatika a výpočetní technika
Téma:	Nástroj pro vizuální modelování cílů. Goal oriented Visual Modeling Tool
Jazyk vypracování:	čeština

Zásady pro vypracování:

Cílem práce je vytvořit nástroj pro modelování požadavků-cílů. Nástroj bude podporovat vizuální modelování a generování dokumentů.

1. Seznamte se s modelováním cílů.
2. Seznamte se s případnými stávajícími nástroji a notacemi pro modelování cílů.
3. Navrhněte a implementujte online nástroj.
4. Vyzkoušejte funkčnost nástroje na ukázkovém příkladu.

Seznam doporučené odborné literatury:

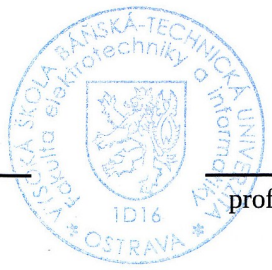
[1] Pfleeger, Shari Lawrence, and Joanne M. Atlee. 2009. Software Engineering: Theory and Practice: Prentice Hall, ISBN 0136061699
[2] Pressman, Roger S. 2010. Software Engineering : A Practitioner's Approach. 7th ed. New York: McGraw-Hill Higher Education, ISBN 9780073375977
[3] Sommerville, Ian. 2010. Software Engineering. 9th ed, International Computer Science Series. Harlow: Addison-Wesley, ISBN 978-0137035151

Další literatura podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Svatopluk Štolfa, Ph.D.**

Datum zadání: 01.09.2017
Datum odevzdání: 30.04.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 30. apríla 2018 .....

Rád by som poďakoval vedúcemu bakalárskej práce **Ing. Svatoplukovi Štolfovi, Ph.D.** za odbornú pomoc a konzultácie pri vytváraní tejto práce.

Abstrakt

Cieľom tejto bakalárskej práce je zoznámenie sa so základnými notáciami pre modelovanie cieľov, ktoré v súčasnej dobe poznáme. Tieto vedomosti nám ďalej pomôžu pri návrhu a následnej implementácii online nástroja na modelovanie cieľov, čo bude predstavovať nástroj s vopred definovanými objektami, ktoré bude možné ďalej modifikovať podľa požiadaviek používateľa. Nástroj bude využívať ASP.NET framework a pre vizuálnu prácu s objektami knižnicu Fabric.js spoločne s Javascriptom. Demonštrácia funkčnosti na konkrétnom príklade bude taktiež zahrnutá v bakalárskej práci.

Kľúčové slová: Modelovanie cieľov, Nástroj, C# , ASP.NET, Fabric.js, Notácie, Javascript

Abstract

The purpose of this bachelor thesis is to get acquainted with the basic notations for currently known goals modeling. This knowledge will further help us in the design and subsequent implementation of the online modeling tool, what will be a tool with predefined objects, that can be further modified according to the user's requirements. The tool will use the ASP.NET framework and for visual work with Fabric.js library it will use objects along with Javascript. Demonstration of functionality on a specific example will also be included in the bachelor thesis.

Key Words: Goals modeling, Tool, C# , ASP.NET, Fabric.js, Notations, Javascript

Obsah

Zoznam použitých skratiek a symbolov

AJAX	–	Asynchronous JavaScript + XML
API	–	Application Programming Interface
CSS	–	Cascade Style Sheet
HTML	–	Hyper Text Markup Language
JS	–	JavaScript
JSON	–	JavaScript Object Notation
MIT	–	Massachusetts Institute of Technology
PDF	–	Portable Document Format
RUP	–	Rational Unified Process
SD	–	Strategic Dependency Model
SR	–	Strategic Rationale Model
SVG	–	Scalable Vector Graphics
UML	–	Unified Modeling Language
VB	–	Visual Basic
WPF	–	Windows Presentation Foundation
XAML	–	Extensible Application Markup Language

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov zdrojového kódu

1 Úvod

Dnešná doba je plná počítačov. Kam len oko dohliadne, nachádzame smartfóny, tablety, notebooky, dokonca aj tie najnenapádnejšie veci ukrývajú v sebe počítač. Ale čo by bol počítač bez softvéru? Pozrime sa na počítač bez softvéru ako na voľnú pláň. Poskytne nám krátky odpočinok, no istotne nám neuľahčí život. Je teda načase, aby prišli páni stavitelia a na pláni postavili niečo užitočné.

Každý projekt, či už ide o stavbu domu alebo softvéru, začína plánom. V oblasti informatiky sú tieto plány tvorené pomocou rôznych notácií, ktoré nám určujú pravidlá pri modelovaní cieľov a teda aj plánov na návrh softvéru. Aplikácia, ktorá dokáže oddeliť tieto notácie, ale zároveň poskytne užívateľovi voľnosť pri tvorbe rôznych diagramov s ľahko naučiteľným ovládaním by narušila koncept súčasne dostupných softvérových riešení. V úvode tejto práce sa preto pozrieme na základné notácie. Ich vlastnosti, spôsob prevedenia, ale aj na ich využitie pri vývoji softvéru. Tento základný prieskum nám pomôže pri programátorskej časti tejto bakalárskej práce. Druhá časť bakalárskej práce bude zameraná na analýzu trhu, čo predstavuje zber informácií o konkurenčných softvérových riešeniach. Táto analýza nám umožní lepší návrh aplikácie tak, aby aplikácia bola user friendly, čo znamená intuitívnu a ľahkú prácu s aplikáciou a jej funkcionalitou. V tretej časti sa pozrieme na návrh samotnej aplikácie a jej uvedenie do činnosti. Taktiež budeme simulovať využitie nástroja na nami zvolenom príklade.

2 Modelovanie cieľov

Súbor inžinierskych požiadaviek, ktorý môže byť použitý pri hlbšej analýze, či už druhu podnikania alebo technologických riešení, sa nazýva modelovanie cieľov. Skúsenosti, ktoré môžeme aplikovať pri modelovaní cieľov závisia priamo od toho, ako jasný a presný cieľ máme pred sebou. V ríši softvérového inžinierstva budú naše ciele vždy uvedené v termíne požadovaných vlastností výsledného softvéru. [14] Vlastnosti týchto cieľov by sa dali zjednodušiť na štyri základné elementy:

- Modifikovateľnosť
- Efektívnosť
- Spoľahlivosť
- Zrozumiteľnosť

2.1 Modifikovateľnosť cieľov

Modifikovateľnosť je v tomto krátkom zozname historicky najstaršia vlastnosť, ktorá vypláva na povrch, keď softvér, ktorý sme navrhli, vyžaduje určité zmeny, či už kvôli hardvérovým požiadavkám v príklade pri prenesení daného softvéru na iné zariadenie. Softvér sa musí vedieť prispôsobiť konkrétnej situácii. Veľkosť obrazovky, jej rozlíšenie, alebo rozloženie procesov do vlákien, aj tieto faktory zavážia pri správnom konštrukčnom riešení. Ďalším problémom môže byť v našom prípade nový operačný systém. Systém, ktorý nám neumožňuje používať niektoré hardvérové prvky, čím znemožní chod programu. V neposlednom rade je to aj z rozmaru klienta. Modifikovateľnosť alebo zmeniteľnosť je tiež potrebná pri výskyte chýb. Tie musia byť odstránené pre správnu funkcionalitu. Modifikovateľnosť nevyžaduje len adaptabilitu softvéru, dobový dizajn, používanie štandardizovaných stavebných blokov softvéru, ladenie pre výkon, ale taktiež je to povinnosť udržiavať softvérové plány a tím, ktorý bude dané požiadavky klientov, ale aj chyby spravovať, čo predstavuje záťaž z finančného hľadiska.

2.2 Efektívnosť cieľov

Efektívnosť je v súčasnej dobe najviac zneužívaná vlastnosť. Prílišná horlivosť a časová tieseň niektorých softvérových firiem presúva túto vlastnosť až na posledné miesto. Najskôr sa rieši funkcionalita, kde môžu byť využité rôzne technické kompromisy. Preto sa otázka efektívnosti rieši až pri testovaní a určite nepatrí k tým dominantným v softvérovom inžinierstve použitých v praxi.

2.3 Spoľahlivosť cieľov

Spoľahlivosť patrí k veľkým základom tohto odvetvia. Zabrániť zhlyhaniu, alebo pádu sofvéru, to sú základné piliere spoľahlivosti. Z finačného hľadiska to predstavuje tú komoditu, kvôli ktorej sa zvýši predajnosť, zisky, či už spokojnosť zákazníkov. Je to niečo, čo sa nedá aplikovať na konci. Sofvér musí byť vyvíjaný spoločne ruka v ruke aj s touto vlastnosťou. Preto spoľahlivosť ma všadeprítomný vplyv na softvérové inžinierstvo v praxi.

2.4 Zrozumiteľnosť cieľov

Zrozumiteľnosť, túto vlastnosť môžeme chápať z viacerých hľadísk. V prvom rade ide o hľadisko použiteľnosti cieľa. Samozrejme záleží, či daný cieľ smerujeme na managment, technických pracovníkov, alebo obyčajných ľudí. Všetky tieto aspekty musia byť zohľadnené nielen pri grafickom riešení, ale aj pri riešení funkčnosti tak, aby každý jeden ťah myši bol ituitívny a priniesol správny efekt. Druhé hľadisko spoznáme vtedy, keď klient zmení svoje pôvodné stanovisko a nás čaká neľahký súboj pri prechádzaní kódu, alebo plánov daného projektu, preto by mala byť zrozumiteľnosť zakomponovaná aj pri samom modelovaní cieľa. Spôsob ako dosiahnuť zrozumiteľnosť modelovania cieľa je zaviesť pri modelovaní pravidlá, ktoré nám presne stanovia ako má jednotlivá úloha vyzerať. Tieto pravidlá alebo notácie ako má správny návrh softvéru vyzerať si rozoberiem v nasledujúcej podkapitole.

3 Notácie pri modelovaní cieľov

Zaviesť pravidlá pri modelovaní cieľov. Dôvod, ktorý radíme na prvé miesto zavedenia pravidiel do modelovania je určite zrozumiteľnosť. Každý aktér, ktorý bude pracovať s modelom ho vie prečítať a porozumieť mu a prípadne ho upraviť alebo rozšíriť tak, aby bol použiteľný v ďalších častiach vývoja softvéru.

3.1 i* / i* framework

Modelovací prístup i* je pokusom o dosiahnutie lepšieho porozumenia systémových procesov tým, že vyberá sociálne koncepty a umiestňuje ich priamo do jadra. Je vhodný najmä pre skorú fázu modelovania systému. Jeho základom je modelovanie organizačných prostredí, v ktorých sa nachádzajú aktéri s rôznymi, častokrát s konkurenčnými cieľmi. Vo veľkej miere preto cieli na aktérov. Toto modelovanie nám pomáha odpovedať na otázku Kto a Prečo [15] i* sa rozdeľuje na dva základné modelovacie komponenty:

- Strategic Dependency Model

  Cieľom modelu SD je zachytiť štruktúru procesov namiesto bežných strategických a technicky procesných modelov činnosti a subjektov. Zachytáva všetko, čo vplýva na aktérov (hercov), ale zároveň nezobrazuje všetko ostatné. Model strategickej závislosti (SD) poskytuje opis procesu z hľadiska siete vzťahov a závislosti medzi aktérmi. Model môže pomôcť identifikovať zainteresované strany, analyzovať príležitosti a zraniteľnosti a rozpoznať príklady vzťahov, ako sú rôzne mechanizmy na zmiernenie zraniteľnosti. Vnútorné ciele nie sú modelované. Skupina uzlov a prepojení v modeli SD tvorí sieť závislostí. Model strategickej závislosti sa snaží modelovať a ďalej prezentovať pohľad na agentov len pomocou ich vonkajších vzťahov. Komplikácie pri tomto druhu modelovania tvoria hlavne rutinné delegácie.

- Strategic Rationale Model

  Model strategického odôvodnenia (SR) poskytuje opis procesov z hľadiska prvkov procesu a príčiny, ktoré sú za nimi. Zatiaľ čo model strategickej závislosti (SD) udržiava úroveň abstrakcie modelovaním iba vonkajších vzťahov medzi aktérmi, model SR upúšťa od abstrakcie tak, aby umožnil celkové pochopenie aktérov ich procesov, ktoré majú byť zobrazené. Model SR opisuje vzťahy, ktoré sú interné pre hercov, napr. vzťahy medzi prostriedkami, a to nám dáva odpoveď na otázky prečo a ako. Pomocou týchto koncepcií modelovania môžeme nachádzať nové procesy, ktoré lepšie riešia ich problém. Model strategického odôvodnenia poskytuje bohatý súbor konceptov pre procesy a dôvody, ktoré sú s nimi spojené.

3.2 KAOS

Predchádzajúca metodika pri modelovaní sa sústredila predovšetkým na subjekty alebo činnosti. Cieľovo orientované metodiky dávajú primárne miesto zámerným cieľom a to vývoj softvérových systémov s entitami a aktivitami, ktoré sú teraz určujúcou identifikáciou cieľov. Zváženie cieľov vedie k skúmaniu alternatívy návrhov systémov - Je možné uviesť cieľ bez toho, aby ste museli určiť, akým spôsobom má byť cieľ dosiahnutý - a tak môže pomôcť projektantovi vybudovať lepší systém. Snáď najúspešnejšou metódou orientovanou na cieľ je KAOS. KAOS uľahčuje identifikáciu ďalších cieľov a ich potreby, objekty, agentov a činnosti systému. KAOS dodáva bohatú ontológiu na zachytenie a modelovanie požiadavky. Jeho meta-úroveň je v podstate taxonómia pojmov, ktoré vedú identifikáciu požiadaviek a ich vzťahov. Ciele sú brané na opísanie očakávaných vlastnosti systému, pričom príkladom je "dosiahnutie zásahu ambulancie". Rámec KAOS žiada analytika, aby definoval túto sieť konceptov pre danú doménu. Ciele sú analyzované identifikovaním vzťahov na cieľ s inými cieľmi grafu. Cieľové grafy sú sémantické siete vzťahov AND / OR / XOR medzi cieľmi, kde je cieľ buď znížený spojením subgoalov alebo (exkluzívne) disjunkcie. Vzťahy medzi cieľmi môžu byť tiež vysledovateľné. [5] Celkovo je špecifikácia KAOS kolekciou nasledujúcich modelov:

- cieľový model, kde sú ciele zastúpené a pridelené agentom
- objektový model, ktorý je UML model, možno odvodiť z formálnych špecifikácií cieľov, pretože sa týkajú objektov alebo ich vlastností
- operačný model, ktorý definuje rôzne služby poskytované softvérovými agentami

Celkovo je KAOS dobre rozvinutá metodológia pre analýzu cieľov orientovaných požiadaviek, ktorá je dodávaná s pevným formálnym riešením. Počas zdokonaľovania cieľov, cieľovej operácie, analýzy prekážok a zmierňovania, KAOS sa spolieha na formálne zdokonaľovanie, ktoré sa raz a navždy osvedčilo. Z tohto dôvodu pri každej aplikácii vzorov používateľ dostane inštančný dôkaz o správnosti zdokonalenia zadarmo. [3]

3.3 Unified Modeling Language

Unified Modeling Language (UML) je štandardný jazyk pre písanie softvérových plánov. UML môže byť použitý na vizualizáciu, špecifikáciu, konštrukciu a dokumentáciu artefaktov softvérového rozsahu. UML je vhodný pre modelovacie systémy od podnikových informačných systémov po distribuovaných webových aplikácií a dokonca aj do pevných vstavaných systémov v reálnom čase. Je to veľmi expresívny jazyk, ktorý sa zaoberá všetkými názormi potrebnými na vývoj a zavedenie takýchto systémov. UML nie je ťažké pochopiť a používať. Naučiť sa uplatňovať UML účinne začína tvorbou koncepčného modelu jazyka, ktorý vyžaduje naučenie sa troch hlavných prvkov: základné stavebné bloky UML, pravidlá, ktoré diktujú ako môžu byť spojené stavebné bloky a niektoré spoločné mechanizmy. [1]. Modelovanie cieľov pomocou UML má bohatú históriu vo všetkých strojárskych odvetviach. Návrhové modely musíme voliť tak, aby mali čo najväčší

vplyv na osvetlenie daného problému. Zle zvolený zdroj nás môže častokrát zmiasť. Prirovnajme tento problém výberu k plánom budovy, kde sú zakreslené okná, dvere, šírky stien, elektroinšalácia, kanalizačný systém a rôzne iné súčasti. Ťažko sa nám bude riešiť problém výpadku elektrickej energie na týchto všeobecných plánoch a bude veľmi nepravdepodobné vyriešenie problému na plánoch od kanalizácie. Preto ak zvolíme plány elektorinštalácie, dostaneme jednoduchý čitateľný a hlavne rýchly spôsob riešenia problému. Vo svete UML preto poznáme tri základné druhy diagramov:

- Štrukturálne diagramy
- Diagramy správania
- Diagramy interakcie

Štrukturálne diagramy v sebe zahŕňajú triedne diagramy, diagramy komponent, diagram nasadenia, diagram balíčkov, diagram inšancií a diagram profilov. Najznámejším predstaviteľom tejto skupiny je triedny diagram. Triedny diagram opisuje statickú štruktúru systému, znázorňuje dátové štruktúry a operácie u objektov a súvislosti medzi objektami. Znázorňuje dátový model systému od konceptuálnej úrovne až po implementáciu. Dátové štruktúry zaraďuje do tried a zobrazuje vzťahy týchto tried. [8]

Diagramy správania popisujú správanie či už jednotlivých aktérov, alebo samotného systému. Najdôležitejšie UML diagram pre modelovanie je diagram činnosti, ktorý teraz podporuje Rational Rose. Schémy aktivít sa často používajú pri vývoji softvéru, projekty na dokumentovanie toku programov, alebo napr. na zobrazenie algoritmu použitého na implementáciu konkrétnej operácie. Aktivitné diagramy majú širokú škálu použití, ktoré dokážu zobraziť činností (postupné a paralelné), spotrebované objekty, používané alebo vyrobené činnosťou, ktorá je zodpovedná za aktivity, vzťahy a závislosti medzi nimi. To všetko je nevyhnutné pri modelovaní. Procesy sú aktívnou súčasťou každej veci vo vývoji softvéru. Popisujú funkcie programu a zahŕňajú prostriedky, ktoré sa používajú, transformujú alebo produkujú. Bežná definícia procesu je (Davenport 1993): Proces je jednoducho štruktúrovaný súbor činností určených na produkovanie špecifického výstupu pre konkrétny softvér alebo zákazníka. Proces je teda špecifickým usporiadaním pracovných činností cez čas a miesto, so začiatkom, koncom a jasne identifikovanými vstupmi a výstupmi. [2]

Diagramy interakcie nám môžu poskytnúť ucelený pohľad nad priebehom jednotlivej interakcie, kde môžeme sledovať napr. tok dát medzi jednotlivými komponentami. Jeden z najznámejších predstaviteľov diagramov interakcie je sekvenčný diagram. Sekvenčné diagramy sú nevyhnutné UML artefakty pre modelovanie správania aspektov systému. Diagramy sú obzvlášť vhodné pre objektovo orientované softvéry, kde predstavujú kontrolný tok počas objektu interakcie. Sekvenčné diagramy zobrazujú súbory interakčných objektov a postupnosti správ, ktoré si medzi sebou vymieňajú. Diagramy môžu tiež obsahovať ďalšie informácie o kontrolnom toku počas interakcie. Môžu to byť podmienky (napríklad äk podmienka c potom poslať správu m

inak poslať správu n ") a opakovanie (napr."poslať správu m viackrát ") alebo závislosť od stavu správania. [13]

3.4 Use-Case Diagram

Centrálna analýza a návrh softvérových požiadaviek je nazývaná ako use-case. Use-case diagramy boli navrhnuté v UML ako notácia pre opis softvérových požiadaviek a správania systému. Slúžili ako komunikačný nástroj medzi vývojármi softvéru a používateľmi. Po rozhovore s potenciálnymi používateľmi softvérového systému, vývojári softvérových požiadaviek by mali zmeniť to, čo používatelia očakávajú v softvérovom systéme - Model požiadaviek, ktorý je daný use-case diagramom. Požiadavky predstavujú to, čo systém očakáva od perspektívy vývojárov softvéru. Použitie diagramov zohráva dôležitú úlohu pri vývoji softvéru. Na jednej strane používatelia softvéru môžu pochopiť, či softvérový systém vyhovuje ich potrebám na začiatku vývoja softvéru, keď vidia použite use-case diagramy. Ich sťažnosti týkajúce sa systému môžu byť priamo hlásené ako požiadavky vývojárom. Použitie use-caase diagramov umožňuje používateľom vyhodnotiť správanie systému pred začatím písania kódu. Na druhej strane použitie use-case diagramov môže byť použité ako plány počas celého vývoja softvéru. Okrem vývojárov môžu požiadavky použiť ďalší vývojári softvéru, ako napríklad dizajnéri modelov a testeri, ktorí môžu ďalej navrhovať a testovať softvérový systém založený na tomto použití use-case diagramov. Väčšina požiadaviek modelov daných use-case diagramom pozostáva z dvoch častí. Jedna je časť diagramu a druhá je textový popis. Diagramová časť uvádza vzťah medzi príkladmi použitia a hercami. Textový opis neformálnej časti predstavuje popis pre každý use-case. Väčšina opisov je písaná v neformálnom jazyku, ako je napr. angličtina. Aj keď UML akceptuje akúkoľvek úroveň formálnosti pre use-case diagramy, vyššia úroveň formálnosti môže zabrániť nedorozumeniam medzi vývojármi softvéru a užívateľmi. [12].

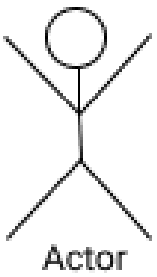
3.4.1 Use-case komponenty

Ako každý diagram aj use-case pozostáva z graficky modelovaných komponent, ktoré majú určitú rolu a funkcionalitu v tomto diagrame. Use-case rozdelíme do 7 komponent vrátane relácií, ktoré môžu medzi danými komponentami nastať:

- Herec(actor)
- Balíček(package)
- Systém(system)
- Use-case
- Dedičnosť(generalization)
- Rozšírenie/Zahrnutie(extends/include)

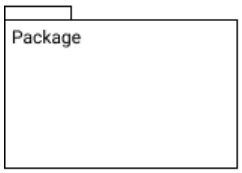
- Asociácia(asociation)

Základnou úlohou herca je predstava role v systéme, čo môžeme voľne chápať ako súbor vlastností, funkcií a právomocí zhrnutých do jednej komponenty. Grafické zobrazenie pozostáva z piktografickej podoby človeka rozšírenej o názov, ktorý sa zvykne umiestňovať pod daný piktogram pozri obrázok 1. Jednou z najčastejších chýb pri používaní tejto komponenty je zlé definovanie funkcií a tým dochádza aj k zdvojeniu jednej role, ktorú zastupujú dvaja, alebo viacerí herci.



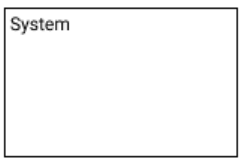
Obr. 1: Herec/Actor

Balíček v sebe združuje viaceré use=case komponenty. Táto vlastnosť nám pomáha riešiť problém veľkého možstva use-case komponent v jednom diagrame. Grafické zobrazenie môžeme definovať ako obdĺžnik s názvom balíčka, nad ktorým sa v ľavom rohu nachádza ďalší menší obdĺžnik pozri obrázok 2. V jednoduchosti sa dá balíček svojou podobou prirovnať zložke v kartotéke.



Obr. 2: Balíček/Package

Aplikácia, podnik, alebo doména, to všetko môže zastávať komponenta s názvom systém. Avšak aj pri tejto komponente si musíme dávať pozor na presné definovanie hraníc daného systému. Hranice sa dajú definovať pomocou názvu systému, ktorý sa nachádza vo vrchnej časti komponenty alebo v slovnom popise. Grafický dizajn teda pozostáva zo spomínaného názvu systému a jedným veľkým obdĺžnikom pozri obrázok 3.

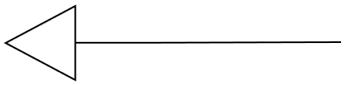


Obr. 3: Systém

Use-case komponenta, po ktorej je pomenovaný celý diagram zastáva úlohu funkcionality v tomto diagrame. Funkcia, ktorú predstavuje musí byť jasne pomenovaná vo vnútri diagramu, avšak musíme si dávať pozor na prílišnú atomizáciu daných úloh, z čoho vzniká problém prílišného preplnenia diagramu. Toto nám môžu pomôcť vyriešiť už predchádzajúce spomínané balíčky. Vizuálne objekt predstavuje elipsa s nacentrovaným textom ako popisom funkcionality pozri obrázok 4.

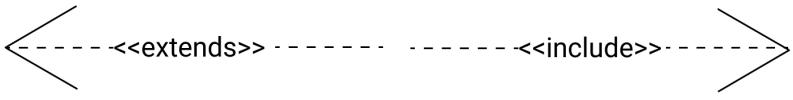
Obr. 4: Use-case komponenta

Dedičnosť alebo generalizácia zvyšuje prehľadnosť diagramu a to z dôvodu odstránenia nadbytočných relácií medzi hercom a use-case komponentou. Ide o vzťah medzi hercami a býva vyjadrený plnou šípkou smerujúcou od herca, ktorý dedí až k rodičovi pozri obrázok 5.



Obr. 5: Generalizácia

Relácie medzi jednotlivými use-case komponentami sú rozšírenie alebo zahrnutie. Pri ich výbere musíme dbať na niekoľko pravidiel. Zahrnutie býva spravidla použité medzi use-case komponentami vtedy, ak základný use-case je nekompletný bez tejto relácie a uvedený prípad je povinný a nie je nepovinný. Relácia zahrnutia pozostáva z jednoduchej šípky smerujúcej k povinnej use-case komponente. Relácia ďalej pozostáva z čiary, ktorá je čiarkovaná a striktného nápisu «include» umiestneného v strede relácie pozri obrázok 6. Druhý prípad relácie použijeme v prípade ak daná komponenta len rozširuje use-case komponentu a nie je tým pádom povinná. Šípka aj grafická podoba relácie je rovnaká až na ten rozdiel, že text sa zmení na «extends» a šípka smerujú k hlavnému use-case komponentu a nie k jeho rozšírujúcej časti pozri obrázok 6. Hlavný dôvod použitia týchto relácií je znovupoužiteľnosť vybraných use-case komponent.



Obr. 6: Extends/Include relácie

Posledným druhomm relácie je obyčajná asociácia. Ide o reláciu, ktorá sa nachádza medzi hercom a use-case komponentou. Je vyjadrená rovnou čiarou medzi komponentami. [10]

4 Nástroje na modelovanie cieľov v kyberpriestore

Svet, do ktorého sme sa narodili je plný ľudí. V niektorých kútoch našej planéty je ich tak veľa, že jediný spôsob ako sa postarať o týchto ľudí je rásť do oblakov. Je nepredstaviteľné, aby otázku vizuálneho modelovania a jeho využitia v kyberpriestore si nikto nepoložil a nevyriešil. Preto sa v tejto kapitole budeme zaoberať hotovými spôsobmi riešenia modelovania cieľov, ich prevedením vizualizáciou funkcionalitou, ale taktiež poplatkami a možnosťou poučenia sa z ich riešenia.

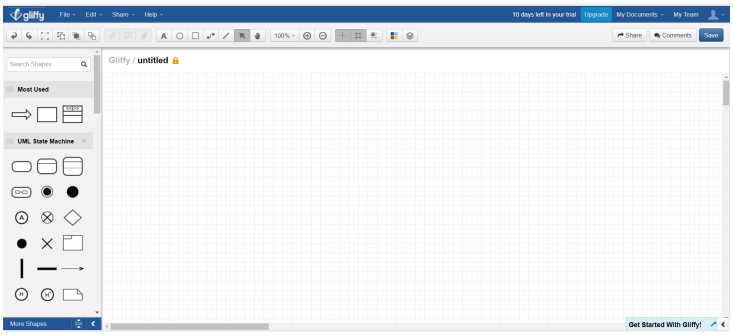
4.1 Gliffy.com

Nástroj, ktorého úlohou nie je len uspokojiť IT časť populácie. Aj takto by sa dal nazvať nástroj, ktorý nájdete pod adresou gliffy.com. Základom tohto šikovného riešenia je cloud-based. Čo môžeme voľne rozumieť ako online zdieľanie softvérového riešenia tak, aby bolo prístupné odvšadiaľ. Mohutné srdce Gliffy bije pod vlajkou HTML5. Práve na túto technológiu firma prestúpila v roku 2012. Bolo však až potom, čo firma dosiahla cieľ jeden milión používateľov. Posledné čísla o rozvoji nástroja však poukazujú na prekročenie hranice troch miliónov. K porovnaniu si môžeme zobrať celú Slovenskú republiku, ktorej počet obyvateľov mierne presiahol päť a pol milióna. Až teraz si môžeme predstaviť obrovský úspech nástroja. Poďme sa však pozrieť na vizualizáciu, funkcionalitu a prácu s ním.

4.1.1 Vizualizácia

V úvode nás privíta pop-up okno s možným výberom typu diagramu. Okno neobsahuje len základné diagramy, ale v zozname môžeme nájsť aj vennové diagramy, wireframe, sieťové diagramy, UML , Mindmap a mnoho ďalších. Rýchly výber a okamžité prepnutie na daný diagram nám ušetrí čas na riešenia problémov pri modelovaní. Zdanie je však klamom. Nech vyberieme akýkoľvek diagram, vždy ľavé bočné menu odkiaľ si vyberáme jednotlivé komponenty ostáva otvorené na Basic Shapes(základných tvaroch). Môžeme povedať, že prvotné pop-up okno nám len strácalo čas a slúžilo viac menej ako reklama vlastného nástroja. Samotné pracovné prostredie je úhľadné. Pozostáva z vrchnej lišty, ktorá je ladená vo farbách firmy a poskytuje nám základné nastavenia nástroja ako ukladanie, import, export, mazanie objektov, ale taktiež môžeme nájsť v jej pravej časti používateľské nastavenia. Aby dizajn lišty nepôsobil preplnene, všetky tieto položky sú schované v drop down menu pozri obrázok 7. Pod lištou nachádzame ďalšiu lištu, ale tentokrát je to akýsi nastavovací panel objektov, ktoré si pridáme do canvasu. Nie všetky nastavenia sú pre každý objekt a tak ostávajú nepoužiteľné nastavovacie panely v takzvanej sivej farbe a bez funkcionality. Samotný canvas sa nachádza pod týmito dvoma lištami a po schovaní ľavej bočnej lišty, v ktorej sa nachádzajú všetky tvary sa roztiahne na celú šírku displeja a tak umožní používateľovi väčší konfort pri práci s diagramom. Je vidno, že nástroj je určený pre použitie výhradne na počítačoch a zariadeniach s veľkým displejom. Pri teste responsibility sa

nám podarilo len zmenšiť canvas a to tým, že sa nám objavila ešte jedna bočná lišta vpravo určená na naše komentáre a tlačidlo na jej opätovné zasunutie prestalo fungovať, čo však nemožno vyčítať firme, keďže tento produkt nie je cielený na podobné zariadenia.



Obr. 7: Gliffy.com vizuálna forma

4.1.2 Práca s nástrojom

Myšlienka pridávania objektov do canvasu je formou drag and drop(chyť a pusti). Po pridaní objektu sa nám pri označenom objekte objaví bublina s rozšírenými možnosťami. Táto bublina nám však poskytuje len základné nastavenie objektu. Nenazvali by sme to ani plus ani mínus, ale osobité riešenie. Relácie medzi objektami sú realizované cez vrchný nastavovací panel a zobrazené hitboxy, do ktorých môže používateľ kliknúť a smerovať svoju reláciu ľubovoľným smerom a k ľubovoľnému objektu. Samozrejme forma šípky alebo relácie bude celkom náhodná a používateľ si ju musí nastaviť v hornej lište. Práca s nástrojom je veľmi intuitívna a nie je potrebné ani úvodné intro pre nových používateľov. Taktiež sa dá veľmi kvitovať možnosť nastavenia atribútov pomocou nastavovacieho panelu.

4.1.3 Klady a zápory nástroja

Medzi hlavné klady tohto nástroja by sme zaradili možnosť ovládania pomocou nastavovacích panelov, čo pri používaní zložitých diagramov je veľmi vhodnou variantou na úpravu diagramu. Taktiež naše ocenenie si zaslúži jednoduchý a prehľadný dizajn, čo dotvára atmosféru nástroja pre všetkých. Čo však túto atmosféru narúša je spoplatnenie služby, aj keď cena pre jednotlivca sa vyšplhala len na 8 euro na mesiac. Pri prvom použití služby je povinná registrácia a je možné si vyskúšať nástroj na 14 dní v trial verzii. Medzi drobné mínusy môžeme zarátať aj zbytočné pop-up menu z úvodu aplikácie. Gliffy si určite zaslúžil svoje miesto medzi velikánmi tohto odvetvia a služby, ktoré toľké roky ponúka, patria medzi tie najlepšie.

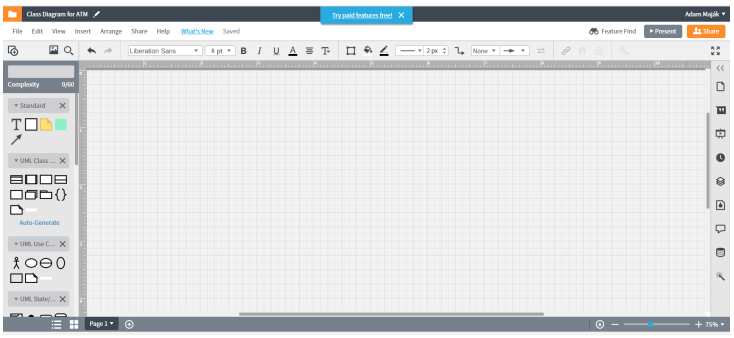
4.2 Lucidchart

Rok 2008 a na trhu sa objavuje nový nástroj na účinné modelovanie diagramov. Firma svojím produktom chcela byť hlavnou konkurenciou Microsoft Visio. A spôsob ako prevalcovať tak

zdatného súpera si našla v online svete. Základy má táto služba postavené na riešení web-based, čo umožňuje zamestnancom, ale aj obyčajným ľuďom pracovať na projekte v reálnom čase tak, aby boli všetky informácie navzájom zdieľané. Firemné programátorské zručnosti firma predviedla v HTML5 a javascripte.

4.2.1 Vizualizácia

Po vizuálnej stránke pozri obrázok8 je nástroj vo veľkej miere podobný. Došlo však zmenám oproti predošlému nástroju a hlavné dve vrchné lišty sa nám tentokrát rozdelili na tri časti, čo znie ako krok k lepšej prehľadnosti. Opak je však pravdou a dizajn líšt pôsobí natesnane a obyčajnému užívateľovi sa dostaví pocit bezmocnosti. Po úvodnom šoku strach rýchlo opadne a intuitívne ovládanie a zameranie na drag and drop udomácni používateľa rýchlosťou blesku. Pribudlo nám ďalšie menu na pravej strane, ktoré svojou funkcionalitou mohlo byť pokojne skryté v hornej lište pod jednou ikonou a reprezentované ako drop down menu. Je vidno, že vývojári si dali prácu aj s responsibilitiou a celý systém pripravili aj na menšie zariadenia ako je napríklad iPad.



Obr. 8: Lucidchart vizuálna forma

4.2.2 Práca s nástrojom

Príjemným prekvapením pri vybratí určitého diagramu je nascrolovanie ľavého menu s komponentami na zvolený diagram. Ostatné funkcionality sa nijako výrazne nelíšia od Gliffy, avšak chýba možnosť nastavovania pomocou doplnkových panelov. Príjemnou funkciou, ktorú môžete využiť je export do pdf súboru, ale taktiež podpora dotykových zariadení.

4.2.3 Klady a zápory nástroja

Pekné a prehľadné vykresľovanie relácií. Podpora dotykových zariadení alebo nascrolovanie ľavého menu v úvode požívania patrí medzi hlavné klady tohto nástroja. Mínusy nie sú tak markantné a skôr ide len o kozmetické úpravy. Občas sa stáva, že používateľ spraví dvojklik nad reláciou, čo vygeneruje textbox nad danou reláciou. Pri jeho odstránení musí byť užívateľ veľmi opatrný, pretože pri intuitívnom stlačení delete sa zmaže celá relácia. Mierne nahustená horná

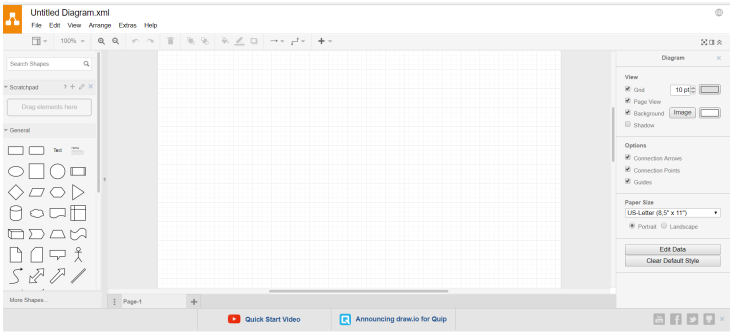
ovládacia lišta a obmedzený počet objektov vo free verzii. Platená verzia stojí 5 euro pre jedného používateľa.

4.3 Draw.io

Pri analýze posledného nástroja sme sa nemohli ubrániť deja vu. Na prvý pohľad vyzerá draw.io, akoby vypadlo z oka lucidchat pozri obrázok 9. O čom môže svedčiť aj používanie HTML5 a javascriptu. Jeden z najmladších, no v súčasnej dobe, jeden z najpoužívanejších nástrojov.

4.3.1 Vizualizácia

Niet sa na čo tešiť, z vizuálnej stránky sa všetky nástroje podobajú ako vajce vajcu. Jednou z najväčších zmien medzi týmito nástrojmi je farba. Čo sa týka rozdielu medzi draw.io a lucidchart, v tom prípade je to len o natočení loga o 90 stupňov. Predsa jednu veľkú zmenu si draw.io odnáša vo svoj prospech a to je pravé bočné menu s doplnkovými nastaveniami ako sme mohli vidieť Gliffy ako malú bublinu. A taktiež responzivita je na vysokej úrovni.



Obr. 9: Draw.io vizuálna forma

4.3.2 Práca s nástrojom

Tu prichádza k vážnejším zmenám a tu je dôvod, čo nakláňa jazýček váh na stranu draw.io. Už len samotné zobrazovanie pravého menu s nastaveniami hra draw.io do kariet. Nesmieme však zabudnúť aj na funkciu importu formátu Gliffy, podpora formátu ISO9001, možnosť rozšírenia o pluginy. Ostatné funkcie sú vo veľkej miere rovnaké.

4.3.3 Klady a zápory nástroja

Jednoznačne medzi kladné stránky patrí free verzia, ktorá obyčajného užívateľa nijak neobmedzuje. Taktiež export do ľubovoľných formátov poskytuje používateľovi veľké pohodlie. Čo sa týka zápornej stránky, pri veľkom počte objektov, ktoré sa navzájom prekrývajú je doslova nemožné si kliknúť a pozmeniť textovú časť daného objektu. Z kompletného testu vzišiel víťaz, ktorý aj napriek svojim malým chybám dokázal premôcť konkurenciu hlavne svojou pridanou hodnotou v oblasti všestrannosti.

5 Vývoj programu

Najťažšie sú vždy začiatky, keď to čo prebýva v našej hlave ako nápad alebo myšlienka potrebujeme pretaviť do výsledného produktu. Zvolenie vhodného postupu vývoja je preto kľúčovým aspektom k dosiahnutiu fungujúce softvéru. Pri návrhu bol využitý vývojový model RUP. Postup vývoja pomocou metódy RUP spočíva v tom, že softvérový produkt je navrhnutý a postavený v postupnosti inkrementálnych iterácií. Každá iterácia obsahuje niektoré alebo väčšinu vývojových disciplín, ako sú požiadavky, analýza, návrh, implementácia a testovanie. To poskytuje disciplínovaný prístup k priraďovaniu úloh a zodpovednosti v rámci vývoja softvéru. Hlavným cieľom RUP je zabezpečiť výrobu vysoko kvalitného softvéru a to splnením potrieb jeho koncových používateľov v rámci predvídateľného rozvrhu a rozpočtu. [7]

5.1 Základné požiadavky

Nástroj, ktorý dokáže modelovať nami vybrané ciele. Aj takto jednoducho by s dali pomenovať základné požiadavky na daný softvér. Ak chceme dosiahnuť čo najväčší dosah na zariadenia s rôznym operačným systémom alebo hardvérovým vybavením, ako najlepšia možnosť sa javí online svet. Pre potreby tejto aplikácie bude potrebné taktiež jednoduché dizajnové riešenie z pohľadu vizualizácie. Ak si odmyslíme vizuálnu stránku môžeme prebrať funkčné nápady z predchádzajúcej kapitoly. Drag and drop je užitočná funkcionalita pri rýchlom zvolení potrebných komponentov. Ďalšou užitočnou funkcionalitou je použitie nastavovacieho panelu tak, aby tvoril akési univerzálne rozhranie pre nastavenie konkrétneho objektu. Ukladanie projektov a jeho opätovné načítanie by v tomto nástroji nemalo chýbať. Objekty, ktoré sa budú nachádzať v oblasti canvasu by mali byť manipulovateľné tak, aby neboli ohrozené základné notácie zvoleného diagramu. Používateľ bude mať možnosť sa prihlásiť a spravovať si svoje projekty, ktoré si predtým uložil do databázy. Hlavná stránka aplikácie by mala obsahovať používateľský manuál pre rýchle riešenia problémov používateľa.

5.2 Vizualizácia

Prvý krok, hneď po zbere požiadaviek, je vytvorenie vizuálneho modelu pre lepšie pochopenie funkcionalít a zároveň rozšírenie stávajúcich požiadaviek na programátorskú časť. Keďže ide o online nástroj, budeme k vizuálnemu modelu pristupovať ako k návrhu webovej stránky. Program, ktorý bol na toto prvotné modelovanie použitý môžeme nájsť pod názvom Figma [1]a umožní nám jednoducho a rýchlo vymodelovať požadované vizualizácie.

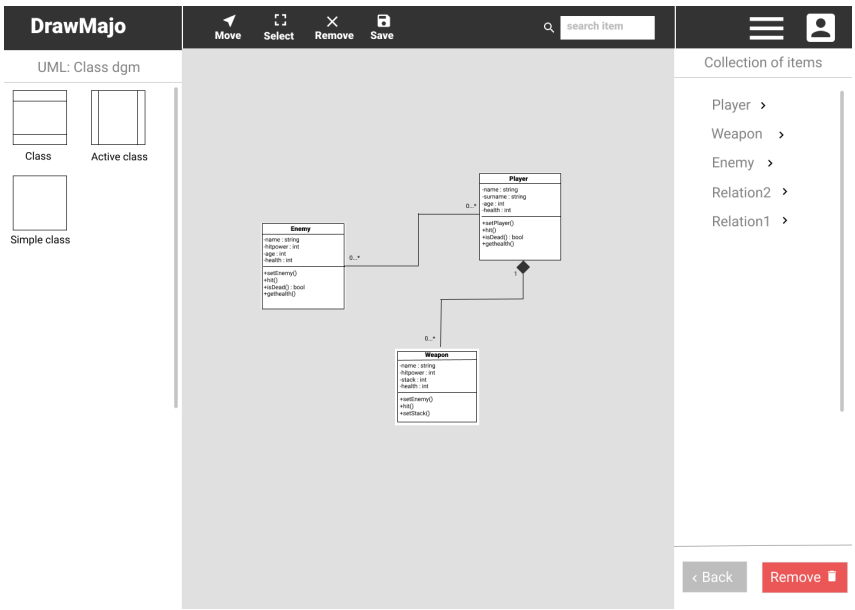
5.2.1 Hlavná stránka

Pri tvorbe hlavnej stánky musíme dbať na jej prehľadnosť. Preto sme zvolili jednoduchú čiernu lištu, do ktorej je vsadené biele menu. Pod hornou lištou je umiestnený názov stránky aj so

základným popisom a možnosťou okamžitého prepnutia sa do modelovacieho nástroja. Ako doplnkové objekty pre úhľadné zobrazenie, ktoré by narušili koncept monotónnosti boli vybrané heslovité popisy nástroja formátované do troch stĺpcov.

5.2.2 Modelovacia časť

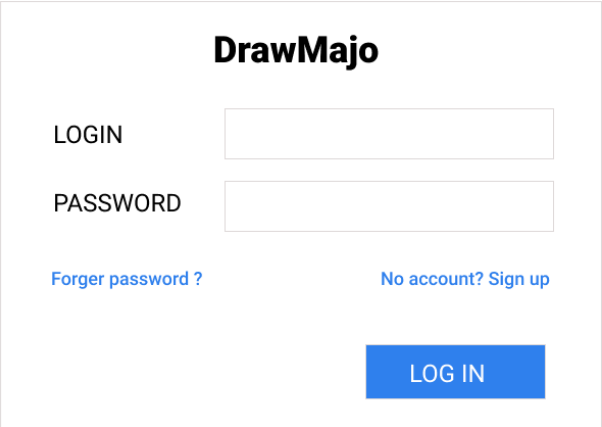
Základ modelovacej časti, ako v predchádzajúcej podkapitole bola horná lišta, ktorá na rozdiel od predchádzajúcej bude rozdelená na tri časti v takom pomere, aby dve krajné časti tvorili predel medzi bočnými lištami, inak povedané šírka pravej a ľavej časti je zhodná so šírkou ľavého a pravého bočného panela. Ľavá časť lišty obsahuje názov a tlačidlo pre zasunutie ľavej bočnej lišty. Stredná časť obsahuje štyri základné tlačidlá pre prácu s modelom a to vyberanie objektov, ich mazanie, ukladanie modelu a na záver vyhľadávanie. Pravá časť obsahuje dve ikony, jedna je na zasunutie prevej bočnej lišty a ikonu pre správu účtu používateľa. Pod touto lištou sa nachádzajú už spomínané bočné lišty. Ľavá bočná lišta je svojím obsahom určená na výber objektov do canvasu, jednotlivé typy objektov sú rozdelené do skupín podľa príslušnosti k danému diagramu. Zobrazenie objektov je pomocou drop down funkcionality. Farebne je lišta ladená v bielej farbe s modrým textom pri aktivácii. Modrá farba bola zvolená ako základná farba stránky s jej unikátnym kódom #3498DB. Nasleduje široký canvas, jeho farba sa mierne líši od bielej, aby dotváral rozdiel medzi lištou a vlastným rozhraním. Selektívne pravé menu pozostáva zo zoznamu objektov, ktoré sa práve nachádzajú na canvase a po rozkliknutí jednotlivého objektu sa nám zobrazí jeho vlastný nastavovací panel. Každý objekt v databáze má jedinečný nastavovací panel. Preto je potrebné, aby dizajn medzi jednotlivými nastavovacími panelmi bol podobný a používateľovi nespôsoboval vrásky pozri obrázok 14. Toto riešenie jednotnosti však ostáva na programátorovi. Základný koncept návrhu sa nachádza na obrázku 10.



Obr. 10: Návrh modelovacej časti

5.2.3 Prihlasovanie a profilová časť

Prihlasovací box je umiestnený na samostatnej stránke a tvorí ho obdĺžnik. Vo vnútri na nachádzajú formulárové boxy pre zadanie emailu a hesla, samozrejme nesmie chýbať ani pomoc pri zabudnutí hesla a odkaz pre vytvorenie nového používateľa. Všetko pod záštitou názvu stránky lokalizovaného v hornej časti obdĺžnika pozri obrázok 11. Profil samotného používateľa je rozdelený na dve časti a to na časť osobných údajov a ďalej na časť, v ktorej sa nachádzajú uložené projekty.



Obr. 11: Návrh prihlasovacieho formulára

5.3 Frontend technológie

Keď chceme postaviť dom, potrebujeme postaviť múry, strechu, osadiť okná a dvere do stien tak, aby sa na nás celý dom nezrútil. Vo svete vývoja webov a webových aplikácií sa táto časť nazýva backend. Frontend môžeme teda voľne prirovnať k pánom maliarom, nábytkárom a dekoratérom. V našom prípade bude dekoratérom samotný používateľ, a preto bude veľká časť tohto softvéru písaná v jazyku určenom pre frontend .

5.3.1 HTML5

HTML5 je vývojom predchádzajúceho kódu HTML, ale aj odpoveďou na zmenu v spôsobe zobrazovania obsahu na webe. Vývojári aplikácií poskytujúci multimediálne a interaktívne služby sa predtým spoliehali na riešenia poskytované tretími stranami, predovšetkým Adobe Flash a v menšej miere, Microsoft Silverlight. HTML5 štandardizuje niektoré z hlavných aspektov, ktoré umožňujú prehliadaču priamo poskytovať tieto funkcie bez potreby inštalácie ďalšieho ovládača alebo doplnku. Hoci HTML5 je samo o sebe štandard, používa sa tiež ako všeobecný pojem pre iné súvisiace technológie ako napr. Kaskádové štýly verzie 3 (CSS3) a JavaScript (JS). Väčšinou sa používa HTML5 pre obsah, CSS3 pre prezentáciu a JS pre definovanie správania ostatných

[1]https://www.figma.com/

dvoch atribútov. [6] V našom prípade riešenia sa bude táto technológia používať pri zobrazení základného obsahu a taktiež ako pomoc pri tvorbe formulárov a ich ošetrení pred zlým vstupom.

5.3.2 CSS

Kaskádové štýly sú jedným z najvýkonnejších nástrojov, ktoré sú k dispozícii pre webových dizajnérov. Väčšina ľudí, ktorí používajú textový procesor na akúkoľvek dobu sa stretnú s princípmi štýlových listov a možno ich použili osobitne štýlovými listami na dosiahnutie konkrétnych výsledkov. Princíp modelu kaskádového štýlu (alebo CSS) sa nelíši. Prostredníctvom štýlu môžete priradiť všetky hodnoty k naším štandardným HTML tagom za účelom ovládania vzhľadu webových stránok. Pomocou štýlov si môžete prevziať kontrolu nad každou časťou vašej webovej stránky: môžete určiť množstvo prázdneho miesta medzi riadkami textu, veľkosť písma, štýl písma, farba pozadia, zarážky, okraje ... [11]. Aktuálna verzia podľa stánky [2]je CSS3, ktorá nám opäť rozširuje funkcionalitu o zaoblené rohy, rôzne transformácie, tiene pri blokových prvkoch a ďalšie drobné vylepšenia oproti predchádzajúcej verzii.

5.3.3 Javascript

Univerzálne jadro jazyka bolo zakotvené v Netscape. Táto verzia jazyka JavaScript na strane klienta umožňuje zahrnúť spustiteľný obsah do webových stránok - to znamená, že webová stránka už nemusí byť len statické HTML, ale môže obsahovať programy, ktoré budú komunikovať s používateľom, ovládať prehliadač a dynamicky vytvárať obsah HTML. Syntakticky je základný jazyk jazyka JavaScript podobný C, C ++ a Java, s programovacou konštruktou, ako je príkaz if, while loop a operátor && . Podobnosť však končí touto syntaktickou podobnosťou. JavaScript je neštandardný jazyk, čo znamená, že premenné nemusia mať špecifikovaný typ. Objekty v JavaScript sú skôr ako Perlove asociačné pole, na rozdiel od štruktúry v C alebo objektoch v jazyku C ++ alebo Java. Objektovo-orientovaný mechanizmus dedenia v jazyku JavaScript je podobný dedeniu z málo známych jazykov Self a NewtonScript. To je úplne odlišné od dedičnosti v jazykoch C ++ a Java. Rovnako ako Perl, JavaScript je interpretovaný jazyk, a tak čerpá inšpiráciu z Perl na mnohých miestach, ako napríklad, jeho regulárny výraz a funkcie ovládania poľa. [4]

5.3.4 JQuery

Podľa [3]je jQuery rýchla a presná JavaScript knižnica, ktorá zjednodušuje prácu v HTML dokumente s udalosťami, manipuláciu, animáciu a Ajaxom pre rýchly vývoj webových aplikácií. jQuery je určený pre zmenu spôsobu, akým píšete JavaScript. jQuery je slobodný a otvorený software pod MIT licenciou. jQuery tiež poskytuje možnosti pre vývojárov na vytváranie pluginov postavených na tejto JavaScript knižnici.

[2]https://www.w3.org/Style/CSS/current-work
[3]http://jquery.com/

5.3.5 Bootstrap

Najpopulárnejšia knižnica komponentov na svete. Bootstrap je open source toolkit pre vývoj s HTML, CSS a JS. Pomocou nej je možné rýchlo prototypovať a vytvárať webové stránky a to vďaka systému mriežky, rozsiahlej zbierke komponentov a výkonnych pluginov postavených na jQuery. [4]

5.3.6 Fabric.js

Fabric.js je silná knižnica naprogramovaná v Javascripte, ktorá umožňuje pracovať s HTML5 canvasom. Fabric poskytuje chýbajúci objektový model pre canvas, ako aj analyzátor SVG, vrstvu interaktivity a celú sadu ďalších nepostrádateľných nástrojov. Je to projekt s úplným otvoreným zdrojom, licencovaný pod MIT, s mnohými príspevkami v priebehu rokov. Tento nástroj našiel uplatnenie na stránke printio [5]– kde umožňuje používateľom navrhnúť vlastné oblečenie. Canvas nám umožňuje v súčasnosti vytvoriť na webe absolútne úžasnú grafiku. Ale uroveň API, ktoré poskytuje, je neuspokojivo nízka. Je to jedna vec, ak chceme jednoducho nakresliť niekoľko základných tvarov na canvase a zabudnúť na ne. Ale akonáhle je potreba akéhokoľvek druhu interakcie, zmena obrazu v ktoromkoľvek bode, alebo kreslenie zložitejších tvarov, situáciu dramaticky mení. Fabric sa snaží tento problém vyriešiť. Natívne metódy canvasu nám umožňujú vypáliť jednoduché grafické príkazy. Chcete nakresliť obdĺžnik? Použite fillRect (ľavá, horná, šírka, výška). Chcete nakresliť čiaru? Použite kombináciu moveTo (ľavá, horná) a lineTo (x, y). Je to ako keby sme maľovali plátno štetcom a vrstva viac a viac oleja na vrchole s veľmi malou kontrolou. Namiesto toho, aby fungoval na takej nízkej úrovni, poskytuje Fabric jednoduchý, ale silný objektový model navyše prirodzených metód. Stará sa o stav plátna a vykresľovanie a umožňuje pracovať s objektami priamo.[6]Fabric disponuje databázou základných objektov, ktoré poskytujú bežnému používateľovi rýchlejšiu prácu pri ich vytváraní. Čo však Fabric neobsahuje pri základných objektoch sú šípky a celkový koncep vytvárania zložitých obrazcov, s ktorými je možná manupulácia, je komplikovaný. Všetky objekty, ktoré sa nenachádzajú v databáze Fabricu sa musia vytvárať pomocou objektu group. Framework samotný poskytuje prácu s objektami ako ich rotáciu, farbu, veľkosť písma, poziciovanie a ponúka aj základné eventy pri zmene, pohybe, označení, alebo rotácii objektov.

5.4 Backend

Stránka webovej aplikácie, ktorá je ukrytá pod povrchom, ale jej vplyv je všadeprítomný. Takto jednoducho by sme mohli definovať backend pri webových aplikáciách

[4]https://getbootstrap.com/
[6]http://fabricjs.com/fabric-intro-part-1
[6]printio.ru

5.4.1 C#

Cieľom C# je poskytnúť jednoduchý, bezpečný, moderný objektovo orientovaný Internetcentric, vysoko výkonný jazyk pre vývoj .NET. C# je teraz úplne zrelý a poučuje sa z poučení získaných počas posledných troch desaťročí. V mnohom, rovnakým spôsobom, ako môžeme vidieť na malých deťoch, ktoré sa učia a preberajú charakteristiky od ich rodičov a starých rodičov. Môžete ľahko vidieť v C# vplyv Java, C++ , Visual Basic (VB) a ďalších jazykov. Základný jazyk C# je odzbrojene jednoduchý a má menej ako 100 kľúčových slov a tucet vstavaných dátových typov, ale je veľmi výrazný, pokiaľ ide o implementáciu moderného programového konceptu. [9]

5.4.2 .Net

Zaujímavou nadstavbou alebo frameworkom nad C# je .Net. Systém. .NET 3.5 predstavuje ďalšie zrenie tohto frameworku a prináša nové spôsoby, ako vytvoriť, dobre skoro všetko.. Stále môžete vytvárať webové aplikácie iba pre servery, ale pomocou AJAX môžete pridať klientov (a AJAX poskytuje podporu oveľa viac, vrátane automatického JSONu kódovanie a dekódovanie). Stále môžete vytvárať aplikácie Windows Forms for Windows, ale môžete tiež vytvárať bohatšie aplikácie Windows pomocou WPF, ktorý používa deklaratívnu syntax nazvanú XAML. Tá istá XAML sa používa pri vytváraní aplikácií WF, ktoré môžu byť používané a okrem iného aj ako obchodná vrstva pre vaše aplikácie. [9]

5.5 Implemenácia nástroja

Zhrnutie všetkých predchádzajúcich bodov nám poskytuje vhodne pripravený pracovný stôl pre implementáciu nástroja. V tejto podkapitole budeme rozoberať problematiku implementácie nástroja a jej riešenie.

5.5.1 Popis a manipulácia s objektami

Objekty, ktoré sa vykresľujú do canvasu fabric.js, sú špeciálne prípady bežných tvarov vykresľovaných pomocou HTML5 a jeho canvasu. Obsah objektov je špeciálny ich atribútmi, ktoré obsahujú. Medzi základné atribúty patrí nepochybne výška, šírka, rotácia alebo uhol natočenia. Pri objektoch, ktoré majú obrysovú čiaru je možné nastaviť aj tento rozmer. Preto naše vlastné prípady musia v konštruktore obsahovať tieto všetky atribúty pre lepšiu manipuláciu a schopnosť vytvorenia objektu. Keďže JavaStript ako hlavný programovací jazyk tohto projektu je objektovo orientovaný, ale beztriedny jazyk pre lepšiu prehľadnosť, budeme používať špeciálne prípady objektov, ktoré obsahujú konštruktor a majú podobné správanie ako trieda, ale neobsahujú deklaráciu premenných. Hlavná konštrukcia vlastného objektu pozostáva zo štyroch základných častí a to sú:

- Konštruktor
- Metóda Draw(canvas)

- Inicializačná metóda InitObject()
- Časť funkčnosti

Samotný konštruktor a základné princípy fungovania by neboli také zvláštne až na ten prípad, že nami zvolený konštruktor obsahuje všetky premenné a atribúty, ktoré sa vyskytnú v celej triede. Z nutnosti istého prehľadu nad kódom a aj budúcej modifikovateľnosti túto počiatočnú definíciu musíme umiestniť sem.

Objekt, ktorý je predávaný do canvasu knižnice Fabric.js musí byť objekt postavený na tejto knižnici a musí byť svojimi atribútmi zhodný s atribútmi tejto knižnice. Preto by konštrukcia takýchto objektov bola zložitá a časová náročnosť projektu by sa rázom strojnásobila podľa našich výpočtov. Preto pri modelovaní vlastných objektov, ktoré nebudú zdieľať kompatibilitu s fabric.js je nutný istý medzikrok a konverzia nášho objektu na fabric.js objektu. Základ objektu pozostáva z množstva objektov fabric.js, ktoré funkcia draw(canvas) preberá a balí do fabric.js objektu s označením Group. Objetku tohto typu sa predávajú všetky objekty fabric.js, ktoré náš objekt obsahuje, ako napríklad obdĺžniky, trojuholníky, textové polia a podobne. Tieto objekty sa premenia na jeden grafický objekt tak, aby následné pozíciovanie v canvase prebiehalo súčasne a objekt si držal svoj tvar. Ako už môžeme tušiť do canvasu sa predáva práve táto jedna premenná s názvom group typu Fabric.Group . Aj z tohto dôvodu musíme riešiť veľký presun dát medzi premennou group a samotným objektom, ktorý danú skupinu vlastní.

Ako už názov metódy môže napovedať, hlavnou úlohou je definovanie základných atribútov objektu. Pri používaní menšieho množstva objektov Fabri.js v jednom vlastnom objekte vzniká problém pozíciovania a samotného prepočtu pozície v rámci nami spomínanej premennej group. Premenná si vo vnútri udržiava vlastný canvas a preto aj pozície menších objektov musia byť relatívne v závislosti od hodnôt premennej group. Táto metóda nám ale nerieši len tento problém, ale zároveň je schopná menšie objekty zväčšovať, zmenšovať, otáčať, alebo úplne vymazať v závislosti nad predom danej notácie objektu.

Časť drobných funkcionalít pozostáva z množstva drobných metód, ktoré sa starajú napríklad o zmenu hodnoty textu prebratú z HTML objektu alebo o definovanie veľkosti poľa. Veľkosťou je táto časť najväčšia a to aj z dôvodu škálovateľnosti a modifikovateľnosti objektov či už ide o zmenu veľkosti písma, hrúbku čiar alebo len samotné pozície objektu v canvase.

5.5.2 Riešenie problému relácií

Relácia predstavuje v našej verzii objekt, ktorý sa svojím zložením nelíši od objektov predstavených v predošlej podkapitole. Fabric.js neposkytuje šípku medzi základnými objektmi, čo pre náš prípad znamená modelovania všetkých šípok zvlášť pomocou poskytovaných objektov. Konštrukčné riešenie však nie je jediný problém. Nutnosť napojenia relácie na objekt aj pri zmenenej pozícii sa riešil následovne. Každý objekt s podporou pripojenia relácie je rozšírený o osem hitboxov. Hitboxy predstavujú prípojné body pre jednotlivé relácie. Tieto body pri ich

vykreslení obsahujú vnútornú polohu v rámci premennej group. Tá musí byť prepočítaná a následne vrátená objektu relácie. Objekt relácie dostal polohu a zvolené id objektu, ktoré si uloží do svojej predom pripravenej premennej pre budúce zaobchádzanie s objektom. Druhá strana relácie môže ostať bez určenia a v tom prípade si zachová pôvodnú pozíciu. Prepočet rotácie objektu relácie je riešený pomocou Pytagorovej vety. Získanie potrebných údajov môžeme zachytiť funkciou recalculateRight, kde môžeme ďalej vidieť aj priebeh prepočtu náklonu.

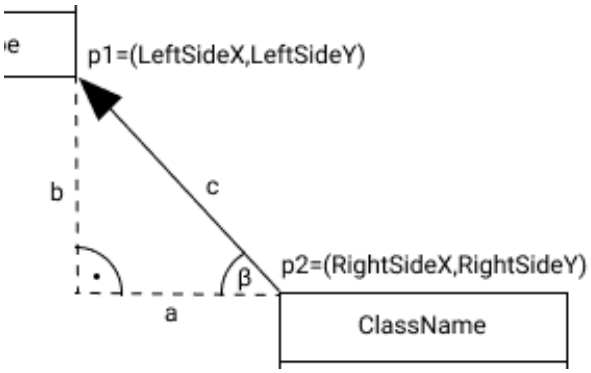
```
 var xOffset = obj.hitBoxArrayCoordinates[this.rightObjectHitbox].x;
 var yOffset = obj.hitBoxArrayCoordinates[this.rightObjectHitbox].y;
 this.rightSideX = obj.previousPosX + xOffset;
 this.rightSideY = obj.previousPosY + yOffset;
var p1 = { x: this.leftSideX, y: this.leftSideY };
var p2 = { x: this.rightSideX, y: this.rightSideY };
var a = p1.x - p2.x;
var b = p1.y - p2.y;
var lenght = Math.sqrt(a * a + b * b);
this.angle = Math.atan2(p2.y - p1.y, p2.x - p1.x) * 180 / Math.PI;
this.width = lenght;
this.previousPosX = this.leftSideX;
this.previousPosY = this.leftSideY;
this.group.setTop(this.leftSideY);
this.group.setLeft(this.leftSideX);
this.initObject();
this.draw(this.canvas);
this.canvas.trigger('object:modified', { target: this.group });
```

Výpis 1: Funkcia recalculateRight pre prepočet relácie

Aby sme lepšie pochopili algoritmus funkcie, zameriame sa na funkcionalitu Fabric.js. Pri voľnom pohľade na problematiku a zhliadnutí dema pre prácu s čiarami bola otázka pozície šípky zobratá na ľahkú váhu. Pretože v našom riešení namiesto čiar používa objekt Fabric.Group, ktorý neumožňuje zadanie vektora na vykreslenie objektu. Práca sa skomplikovala. Zadanie pozície a následne natočenie objektu taktiež nie je možné, čo Fabric.js znemožnil chýbajúcou funkciou rotácie okolo začiatočného alebo koncového bodu. Možná je teda rotácia objektu okolo stredu. Výsledný návrh pozostáva teda z vytvorenia pomyselného pravoúhleho trojuholníka pozri obrázok 12 , kde krajné body prepony sú dané pozíciou hitboxu, čo nám poskytne znalosť dvoch odvesien trojuholníka a jedného pravého uhla. Dopočet natočenia, čiže úhla beta v trojuholíku prebiehal následujúcim vzorcom :

$$\beta = \frac{\frac{p2.y - p1.y}{p2.x - p1.x} * 180}{\pi} \tag{1}$$

Keď máme uhol beta vypočítaný, stačí ho nastaviť ako parameter angle do premennej group. Nechať celú skupinu prepočítať a opätovne nastaviť pozíciu hitboxu, na ktorý je daná relácia pripojená.



Obr. 12: Vizuálne zobrazenie pomyselného trojuholníka

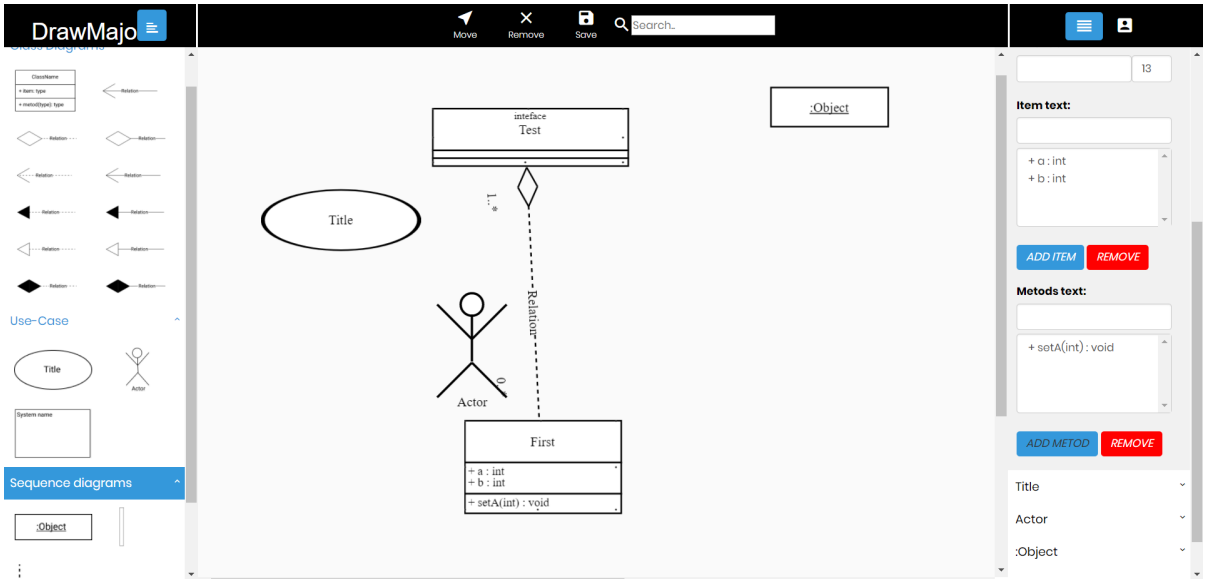
5.5.3 Vizuálna stránka nástroja

Na opísané vizuálne vlastnosti a aj grafický návrh z predchádzajúcej kapitoly je potrebné natiahnuť kostru, ktorá pozostáva z HTML5, CSS, a JavaScriptu, ako všestranný pomocník je využitý Bootstrap. Základ stránok sa odvíja od formátu cshtml - html dokument, v ktorom je možné využívať funkcionalitu .Net technológie. Viditeľné ikony sú prevzaté zo stránky Material-Designe [7]Už pridaným ikonám sem doplnili triedu a pomocou CSS aj formát. Mriežka využitá z Bootstrap sa používa až pod nastavovacou lištou, čo predstavuje značné zjednodušenie responsibility. Ľavý aj pravý bočný panel je konštruovaný formou Side Navigation, ktorú možno nájsť na stránkach w3schools. Drop down menu bolo pridané do týchto panelov samostatne. Ľavý panel sa radí do statických kvôli svojmu nemennému obsahu. Pri pravom paneli bola požiadavka na dynamické zmeny počas behu stránky. Zmena položiek menu, aj samotného obsahu daných položiek je riešená formou preddefinovaných vzorových dokumentov. Realizácia vzorov sa odohráva cez HTML form – formulár. Táto časť HTML prestavuje potrebné atribúty ako textové polia, polia pre validáciu zadaného čísla, select tag pre výber z možností. Každý dokument odpovedá jednému typu objektu.

```
  $.ajax({
url: "../../Content/Pattern/Relation.html",
type: "get",
async: false,
success: function (data) {
    $("#rightMenuScrollInsert").append(data);
}
```

Výpis 2: Funkcia pre načítanie vzoru HTML dokumentu

Pri umiestnení objektu do canvasu sa volá funkcia na pridanie položky do pravého menu. Vyberie sa vhodný html vzor a importuje sa cez jQuery funkcie na stránku. Toto riešenie je rozšírené o prídavnú funkciu na zmenu všetkých id atribútov v novo pridanom kuse html kódu. Id časti kódu sú volené tak, aby každá manipulovateľná časť mala svoje vlastné a jedinečné id, ktoré pozostáva z názvu html tágu a id objektu, ktorý dané nastavovacie menu vlastní. Id objektu je jedinečné v celom canvase a preto je zhoda id html atribútov vylúčená. Hotová vizualizácia je k nahliadnutiu na obrázku 13.



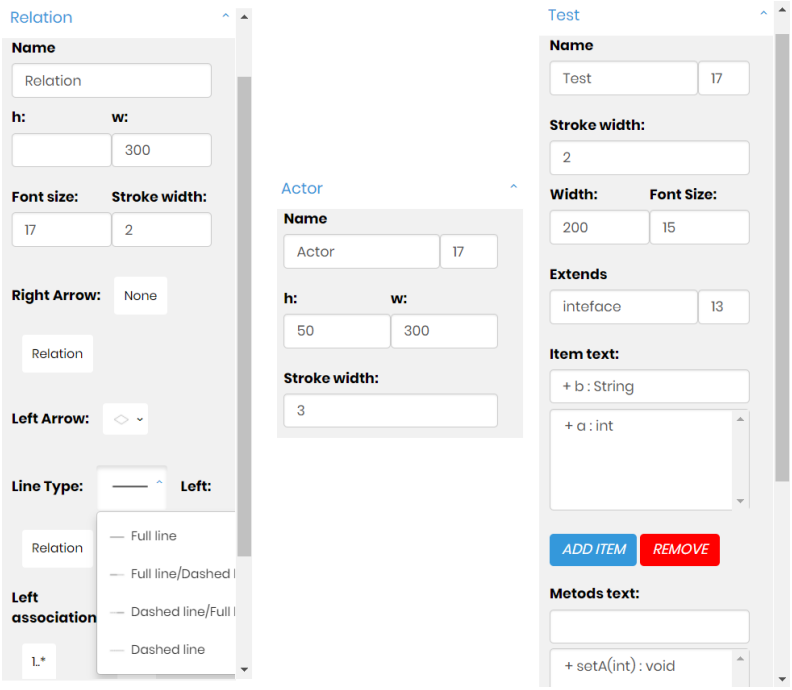
Obr. 13: Výsledné vyhotovenie vizualizácie

5.5.4 Ovládateľnosť

Štruktúry starajúce sa o schopnosť ovládania objektov v canvase sú prevzaté od knižnice Fabric.js. To sa týka pohybu objektov po canvase spôsobom drag and drop . To čo spôsobuje problém pri preberaní tejto funkcionality je vzájomné prepojenie objektov Fabric.js a nášho vlastného globálneho objektu. Vo všeobecnosti máme v programovej časti vytvorené pole, v ktorom sú naše objekty. Na predanie objektov do canvasu používame cyklus, kde daný objekt načítame a predáme do canvasu len objektovú premennú group. Avšak pri pohybe sa táto zmena nepredáva naspäť do objektu a v tom prípade pri zmene v nastavovacom paneli je objekt vykreslený na pôvodných súradniciach. Aplikácia canvas udalosti na celý canvas nám umožní v čase zmeny správne identifikovať objekt a predať dáta do nášho objektu. Samotné vykresľovanie podľa želania užívateľa je tvorené sadou nastavovacích panelov, ktoré sa nachádzajú v pravom menu. Vo všeobecnosti panely obsahujú základné nastavenia, ktoré sú pre všetky objekty skoro rovnaké, ale taktiež obsahujú rozšírenú štruktúru, ktorá bude využitá pri manipulácii s objektom, ktorý má presne vymedzené notácie a ovládanie, napríklad pri zápise metód do triedneho diagramu je

[7] https://material.io/icons/

táto funkcionalita zastúpená pridávacím boxom a zoznamom pridaných atribútov. Dané atribúty môžeme v ďalších krokoch pridávať a mazať. Obrázok 14 ukazuje druhy ovládacích panelov.



Obr. 14: Nastavovacie panely pre rôzne druhy objektov

Ovládateľnosť relácií je koncentrovaná v pravej časti nastavovacieho menu. V sekcii konkrétnej relácie je možné si zvoliť v dropdown liste rôzne druhy nastavenia od typu čiar, ktoré môžu byť plné, čiarkované alebo na polovicu plné až po druh šípky, ktorý daná relácia má mať. Aplikácia vzťahu medzi dvomi objektami sa rieši v dropdownli list z výberu objektov, ktoré majú byť na pravej alebo ľavej strane. Ku zvoleným objektom sa relácia uzamkne a pomocou predom spomínanej funkcie sa prekresľuje spoločne s nimi. Pri výbere objektu má používateľ možnosť zvoliť jeden z ôsmych hitboxov, ktoré objekty obsahujú a tak lepšie manipulovať s pozíciou šípky. Samozrejmou vecou pri uzamknutí k objektu je jej nemožnosť manipulácie.

5.5.5 Riešenie drobných funkcionalít

Funkcie, ktoré svojím rozsahom nepatria k tým najväčším ani najzložitejším, ale význam je skrytý v ich jednoduchosti a ľahkosti pri používaní a bez niektorých zmienených funkcií, by pocit z aplikácie bol značné narušený a neúplný. Medzi tie hlavné ovplyvňujúce funkcie sme vybrali následovné štyri:

- Tvorba objektov v canvase - Výber objektu z ľavého menu a potiahnutie nad canvas v sebe skrýva funkciu, ktorá ostáva bežnému pozorovateľovi ukrytá. Funkcionalita vykreslenia je značne komplikovaná z dôvodu zmeny pozadia, nad ktorým sa kurzor presúva počas

držania. Zvolený postup je preto nasledovný. Všetky elementy v ľavom menu majú nastavenú hodnotu draggable na true, pred ktorou sa nachádza jedinečné id. Nad všetky id v ľavom menu je umiestnený eventlisener a v prípade pretiahnutia objektu sa zavolá funkcia add() pre objekt, ktorý bol potiahnutý. Pozíciu si konštruktor vyberá z pozície kurzora v momente, keď bol daný objekt pustený, plus k týmto hodnotám sú odpočítavané hodnoty pozície canvasu na stránke pre presne zameranie. Vo funkcii add() sa postupne volajú funkcie initObjekt() a draw(canvas), ktoré predstavujú už samotné vykreslenie objektu.

- Zmena hodnoty objektu cez nastavovacie menu- Zaobchádzanie s objektami canvasu je z väčšej miery ovládané z pravého menu. Všeobecne pri tvorbe pravého nastavovacieho menu platí, že každý riadok alebo záznam v tomto drop down menu patrí práve jednému objektu, a teda aj nastavenia, ktoré sa vyberajú, majú presne stanovené id podľa objektu, ktorému patria. Toto id sa odosiela ako parameter pri zavolaní funkcie onchange. Id však nie je v tomto prípade číslo, ale skladá sa aj z písma, čo rieši funkcia na získanie id z takéhoto reťazca.Pomocou jQuery sa vyberie hodnota z poľa a umiestni sa do premennej objektu pomocou jej vlastnej funkcie.

- Ukladanie vykresľovacej plochy- Projekty, na ktorých sme dlho pracovali sa dajú jednoducho uložiť a načítať pomocou funkcie, ktorá nám umožní zabaliť všetky objekty do txt súboru a neskoršie opätovné načítanie. Všetky objekty ktoré máme v canvasu pochádzajú z pola canvasItems. Toto pole obsahuje celý nami vytvorený objekt, či už ide o naše premenné a funkcie ale aj objekty Fabric.js. Tu nastáva problém, pretože objekty Fabric.js v sebe nesú referenciu na canvas a samotný canvas obsahuje refenenciu na objekty. Využili sme mapper funkciu, ktorá nám vytvorí objekt s názvom mapper a obsahuje všetky základné hodnoty pre modelovanie daného objektu v canvase, ale bez objektov Fabric.js. Takto vytvorený objekt umiestňujeme do poľa a hotové pole so všetkými obejktami prechádzame funkciou stringify. Vysledokom je string, s ktorým môžme pracovať podľa vlastného uváženia. V našej aplikácií je je možný export do dvoch formátov a to formát PNG a formát DrawMajo. Samotný export do png podporuje Fabric.js a tak sa dá táto operacia zvládnuť pomocou par riadkov kódu. Súbor DrawMajo Je nami predom zmienený textový subor na opätovné načitanie.

- Načítanie objektu typu Relation – Relácia patrí medzi najzložitejšie objekty tohto nástroja. Pri načítaní konkrétnej relácie musí byť vzatý do úvahy každý typ relácie. Preto všetky relácie pozostávajú zo šesť miestného kódu. Kód je delený do troch časti a to ľavá šípka, stredová čiara a pravá šípka. Pomocou tohto kódu dokáže funkcia presne určiť druh šípky a určiť jej správny tvar.

5.6 Test konkretného príkladu

Testovacia časť bude spočívať v nasimulovaní modelovania v našom vytvorenom nástroji. Náhodne vytvorené zadanie nám poskytne reálnu spätnú väzbu pri ďalšom rozvoji nástroja, taktiež nám môže pomôcť odhaliť chyby, ktoré pri programovaní vznikli. Nástroj podporuje viacero typov diagramov ako je triedny diagram, aktivitný diagram, usecase diagram, ale aj exotickejšie diagramy i*, kaos. Pretože v tejto práci sme sa snažil prideliť viacej priestoru usecase diagramom, preto aj náš testovací príklad bude z tohoto prostredia.

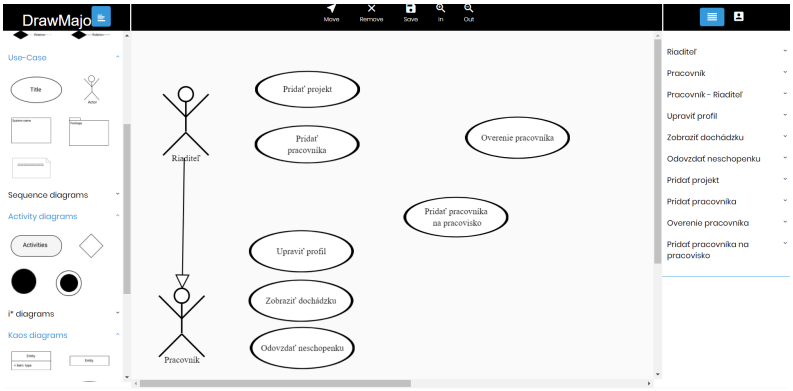
5.6.1 Zadanie príkladu

Príklad bude simulovať firemné prostredie. Základný systém obsahuje dvoch aktérov a to riaditeľa a pracovníka. Riaditeľ špeciálnym prípadom pracovníka. Pracovník má možnosť v systéme upraviť si profil, zobraziť si svoju dochádzku a v prípade neprítomnosti odovzdať neschopenku. Riaditeľ je svojou formou najrozšírenejší prípad pracovníka. Dokáže pridávať projekt, povyšovať pracovníkov. Pridanie nového pracovníka vyžaduje overenie údajov zamestnanca a zároveň pri pridávaní dokáže riaditeľ pridať zamestnanca aj na požadované pracovisko.

5.6.2 Postup riešenia

Ako prvé pri riešení tejto úlohy sme si pridali aktérov, ktorých sme medzi sebou prepojili pomocou relácií v pravom nastavovacom menu. Formácia hitboxov je zobrazená v manuálovných stránkach. Ďalším krokom bolo pridanie všetkých usecase objektov, ktoré sa v schéme nachádzajú, prepojili sme ich reláciami. Postup je možné vidieť na obrázku 15 a na obázku 16

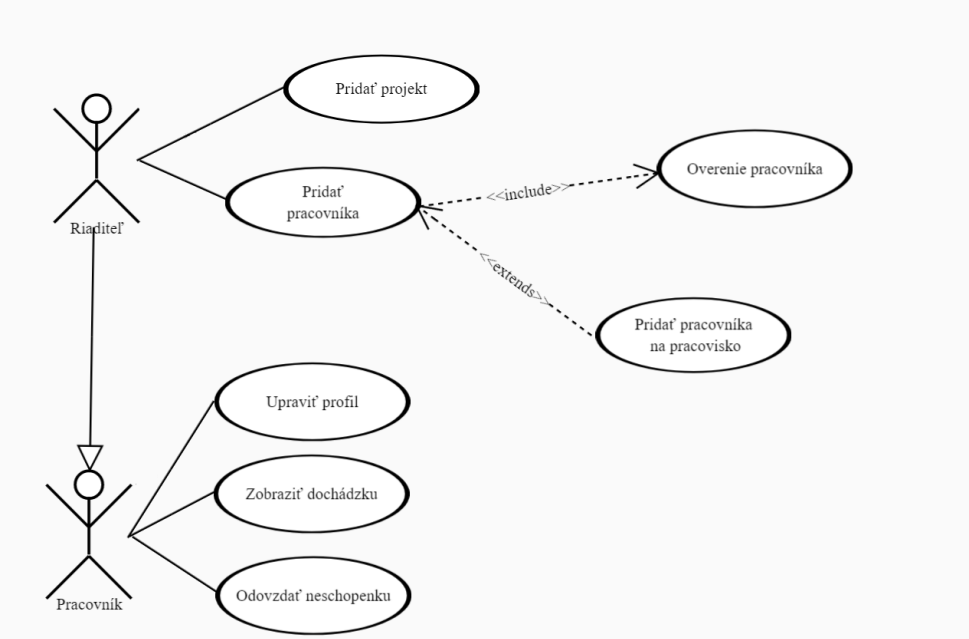
Obr. 15: Prvý modelovací krok



Obr. 16: Druhý modelovací krok

5.6.3 Hotový výstup

Hotové riešenie je k nahliadnutiu v subere s názvom test a taktiež na obrázku17. Formáty v prílohe sú png a dwmj(DrawMajo).



Obr. 17: Hotové riešenie príkladu

6 Záver

Cieľom tejto bakalárskej práce bolo zoznámiť sa s notáciami modelovacích jazykov ako KAOS, i*, UML diagramami a predovšetkým s Use-Case diagramom. Čo bolo splnené a následne využité v ďalšom cieli. Ako ďalší cieľ práce bola určená analýza nástrojov na modelovanie cieľov. Predstavovalo to výber jedných z najpoužívanejších nástrojov, ktoré sú dostupné online. Vybrané boli Gliffy.com, Lucidchart a Draw.io. Táto analýza nám poskytla potrebné podklady v oblasti vizualizácie funkcionality, ale aj ovládateľnosť, čo nám pomôže pri vývoji vlastnej aplikácie. Súčasne sa nám podarilo splniť cieľ, ktorý pozostával z vývoja nástroja. Nástroj bol postavený na Javascripte a HTML5 ako pomocníka s prácou s canvasom bola vybratá knižnica Fabric.js. Všetky tieto prvky nám pomohli vytvoriť nástroj ktorý dokáže modelovať rôzne druhy diagramov s podporou notácií, ktoré sme zmienili v predchádzajúcom cieli. Posledným cieľom bolo nástroj otestovať na reálnom príklade. Výsledky samotného cieľa a taktiež postup pri riešení je popísaný v podkapitole 5.6. Tento test prebehol úspešne a pozostával v modelovaní Use-case diagramu, ktorý je uložený v prílohe s názvom test. V tabuľke 1 nájdeme porovnanie všetkých zmienených modelovacích programov vrátane nášho riešenia.

	LucidChart	Gliffy.com	Draw.io	DrawMajo
Ukladanie do PDF	Áno	Nie	Áno	Nie
Visio Import	Áno	Áno	Áno	Nie
Priama podpora i*	Nie	Nie	Nie	Áno
Priama podpora KAOS	Nie	Nie	Nie	Áno
Cloud ukladanie	Áno	Áno	Nie	Áno
Voľná licencia	Nie	Nie	Áno	Áno
Možnosť modelovania ľubovoľných objektov	Áno	Áno	Áno	Nie
Resposibilný dizajn	Áno	Nie	Áno	Áno
Chrome App	Nie	Nie	Áno	Nie
Napájanie relácií v canvase	Áno	Áno	Áno	Nie

Tabuľka 1: Porovnanie online nástrojov

V ďalšej vývojovej fáze tohto nástroja by bolo možné sa zamerať na presuny riešení medzi nástrojom a riešením napr. Visio alebo Gliffy. Pre budúcnosť by bolo dobre prepracovať systém ovládania tak, aby bol lepšie adaptabilný pre nových používateľov. To by spočívalo v prispôsobení sa konkurencii. Aplikácia je voľne dostupná na[8].

[8] drawmajo.aspfree.cz

Literatúra

[1] Grady Booch. *The unified modeling language user guide*. Pearson Education India, 2005.

[2] Hans-Erik Eriksson and Magnus Penker. Business modeling with uml. *New York*, pages 1–12, 2000.

[3] Patrícia Espada, Miguel Goulão, and João Araújo. A framework to evaluate complexity and completeness of kaos goal models. In *International Conference on Advanced Information Systems Engineering*, pages 562–577. Springer, 2013.

[4] David Flanagan. *JavaScript: the definitive guide*. Ö'Reilly Media, Inc.", 2006.

[5] William Heaven and Anthony Finkelstein. Uml profile to support requirements engineering with kaos. *IEE Proceedings-Software*, 151(1):10–27, 2004.

[6] Antero Juntunen, Eetu Jalonen, and Sakari Luukkainen. Html 5 in mobile devices–drivers and restraints. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 1053–1062. IEEE, 2013.

[7] Asif Irshad Khan, Rizwan Jameel Qurashi, and Usman Ali Khan. A comprehensive study of commonly practiced heavy and light weight software methodologies. *arXiv preprint arXiv:1111.3001*, 2011.

[8] Helena Kučerová. Metodiky ontologického inženỳrství. *Ikaros: elektronickỳ časopis o informační společnosti*, 15(5), 2011.

[9] Jesse Liberty. *Programming C#: Building. NET Applications with C*. Ö'Reilly Media, Inc.", 2005.

[10] Susan Lilly. Use case pitfalls: top 10 problems from real projects using use cases. In *Technology of Object-Oriented Languages and Systems, 1999. TOOLS 30 Proceedings*, pages 174–183. IEEE, 1999.

[11] Duncan Reed and Peter J Thomas. Cascading style sheets. In *Essential HTML fast*, pages 111–121. Springer, 1998.

[12] Wuwei Shen and Shaoying Liu. Formalization, testing and execution of a use case diagram. In *International Conference on Formal Engineering Methods*, pages 68–85. Springer, 2003.

[13] Santosh Kumar Swain, Durga Prasad Mohapatra, and Rajib Mall. Test case generation based on use case and sequence diagram. *International Journal of Software Engineering*, 3(2):21–52, 2010.

[14] Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249–262. IEEE, 2001.

[15] Eric Yu and John Mylopoulos. Why goal-oriented requirements engineering. In *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality*, volume 15, pages 15–22, 1998.

Adresárová štruktúra príloženého disku

- Program - Obsahuje komplentý projekt pre nasadenie na web spracovaný v programe Visual Studio
- Weby - Obsahuje stiahnuté kópie webových stránok použitých v tejto bakalárskej práci
- Test - Obsahuje testovací priklad v png a dwmj
- Manuál - Obsahuje manuál pre prácu s nástrojom